2802Task3Operator

May 12, 2025

0.1 Arthematic Operator in Python

0.1.1 Integers

[4]:
```
print('Addition:',1+3)
```

```
Addition: 4
```

[5]:
```
print('Substraction:',4-1)
```

```
Substraction: 3
```

[7]:
```
print('ABC:',1*3)
```

```
ABC: 3
```

[9]:
```
print('ABC:',2+2)
```

```
ABC: 4
```

[10]:
```
print('Division:',9/3)
```

```
Division: 3.0
```

[11]:
```
print('Division:',9//3) without decimal
```

```
Division: 3
```

[12]:
```
print('Division:',10/3)
```

```
Division: 3.3333333333333335
```

[14]:
```
print('Modulus:',4/3) ##Gives reminder
```

```
Modulus: 1.3333333333333333
```

[15]:
```
print('Modulus:',4//3) gives remainder without decimal
```

```
Modulus: 1
```

[16]:
```
print('Exponential:',3**2) 3power2
```

```
Exponential: 9
```

[17]:
```
print('Exponential:',3**3)3power3
```

```
Exponential: 27
```

0.1.2 Floating Numbers

[20]:
```
print('Floating Number,PI',3.14)
```

```
Floating Number,PI 3.14
```

[19]:
```
print('Floating Number,gravity',9.18)
```

```
Floating Number,gravity 9.18
```

0.1.3 Complex Numbers

[21]:
```
print('Complex numbers:',1+1j)
```

```
Complex numbers: (1+1j)
```

[22]:
```
print('Multiplying Complex numbers:',(1+1j)*(1-1j))
```

```
Multiplying Complex numbers: (2+0j)
```

[23]:
```
print('Multiplying Complex numbers:',(1+2j)*(1-1j))
```

```
Multiplying Complex numbers: (3+1j)
```

0.1.4 Declaring the Variable at the top first

[24]:
```
a = 3 # a is a variable name and 3 is an integer data type
b = 2 # b is a variable name and 3 is an integer data type
```

[25]:
```
# Arithmetic operations and assigning the result to a variable
```

[29]:
```
total=a+b
diff=a-b
product=a*b
division=a/b
remainder=a%b
floor_division=a//b
exponential=a**b

# Sum is built-in function try to avoid overriding built-in functions
```

```
print(total)
print('a+b=',total)
print('a-b=',diff)
print('a*b=',product)
print('a/b=',division)
print('a%b=',remainder)
print('a//b=',floor_division)
print('a**b=',exponential)
```

```
5
a+b= 5
a-b= 1
a*b= 6
a/b= 1.5
a%b= 1
a//b= 1
a**b= 9
```

0.1.5 Declaring values and organizing them together

[48]:
```
num_one=8
num_two=4
```

[49]:
```
# Arithmetic operations
total = num_one + num_two
diff = num_two - num_one
product = num_one * num_two
div = num_two / num_one
remainder = num_two % num_one
```

[50]:
```
# Printing values with label
print('Total:',total)
print('Difference:',diff)
print('Product:',product)
print('Division:',div)
print('Remainder:',remainder)
```

```
Total: 12
Difference: -4
Product: 32
Division: 0.5
Remainder: 4
```

0.1.6 Calculating area of a circle

[53]:
```
radius = 10                                    # radius of a circle
area_of_circle=3.14 * radius ** 2          # two * sign means exponent or power
```

[54]:
```
print('Area of a circle:', area_of_circle)
```

```
Area of a circle: 314.0
```

0.1.7 Calculating area of a rectangle

[55]:
```
length = 10
width = 20
area_of_rectangle = length * width
```

[56]:
```
print('Area of Rectangle:',area_of_rectangle)
```

```
Area of Rectangle: 200
```

0.1.8 Calculating a weight of an object

[59]:
```
mass = 75
gravity = 9.81
weight = mass * gravity
print(weight,'N')
```

```
735.75 N
```

[60]:
```
print(3>2) #True, because 3 is greater than 2
```

```
True
```

[61]:
```
print(3>=2)
```

```
True
```

[62]:
```
print(3<2)
```

```
False
```

[63]:
```
print(2<3)
```

```
True
```

[64]:
```
print(2<=3)
```

```
True
```

[65]:
```
print(3==2)
```

```
False
```

[67]:
```
print(3!=2)
```

```
True
```

[69]:
```
print(len('mango')==len('orange'))
```

```
False
```

[70]:
```
print(len('mango')!=len('orange'))
```

```
True
```

[71]:
```
print(len('mango')<len('orange'))
```

```
True
```

[72]:
```
print(len('mango')>len('orange'))
```

```
False
```

[73]:
```
print(len('milk')==len('meat'))
```

```
True
```

[74]:
```
print(len('tomato')==len('potato'))
```

```
True
```

0.1.9 Boolean comparison

[75]:
```
print('True==True:',True==True)
```

```
True==True: True
```

[76]:
```
print('True==False:',True==False)
```

```
True==False: False
```

[77]:
```
print('False==False:',False==False)
```

```
False==False: True
```

[78]:
```
print('True and True:',True and True)
```

```
True and True: True
```

[79]:
```
print('True and False:',True and False)
```

```
True and False: False
```

0.1.10 Another way comparison

[80]:
```
print('1 is 1', 1 is 1)                    # True - because the data values are␣
 ↪the same
print('1 is not 2', 1 is not 2)            # True - because 1 is not 2
print('A in Asabeneh', 'A' in 'Asabeneh') # True - A found in the string
print('B in Asabeneh', 'B' in 'Asabeneh') # False -there is no uppercase B
print('coding' in 'coding for all') # True - because coding for all has the␣
 ↪word coding
print('a in an:', 'a' in 'an')      # True
print('4 is 2 ** 2:', 4 is 2 ** 2)   # True
```

```
1 is 1 True
1 is not 2 True
A in Asabeneh True
B in Asabeneh False
True
a in an: True
4 is 2 ** 2: True
```

```
<>:1: SyntaxWarning: "is" with a literal. Did you mean "=="?
<>:2: SyntaxWarning: "is not" with a literal. Did you mean "!="?
<>:7: SyntaxWarning: "is" with a literal. Did you mean "=="?
<>:1: SyntaxWarning: "is" with a literal. Did you mean "=="?
<>:2: SyntaxWarning: "is not" with a literal. Did you mean "!="?
<>:7: SyntaxWarning: "is" with a literal. Did you mean "=="?
/tmp/ipykernel_1517/2099784577.py:1: SyntaxWarning: "is" with a literal. Did you
mean "=="?
  print('1 is 1', 1 is 1)                    # True - because the data values are
the same
/tmp/ipykernel_1517/2099784577.py:2: SyntaxWarning: "is not" with a literal. Did
you mean "!="?
  print('1 is not 2', 1 is not 2)            # True - because 1 is not 2
/tmp/ipykernel_1517/2099784577.py:7: SyntaxWarning: "is" with a literal. Did you
mean "=="?
  print('4 is 2 ** 2:', 4 is 2 ** 2)   # True
```

[81]:
```
print(3 > 2 and 4 > 3) # True - because both statements are true
print(3 > 2 and 4 < 3) # False - because the second statement is false
print(3 < 2 and 4 < 3) # False - because both statements are false
print(3 > 2 or 4 > 3)  # True - because both statements are true
print(3 > 2 or 4 < 3)  # True - because one of the statement is true
print(3 < 2 or 4 < 3)  # False - because both statements are false
print(not 3 > 2)     # False - because 3 > 2 is true, then not True gives False
print(not True)      # False - Negation, the not operator turns true to false
print(not False)     # True
```

```
print(not not True)  # True
print(not not False) # False
```

```
True
False
False
True
True
False
False
False
True
True
False
```

[82]:
```
print(True*2)
```

```
2
```

[84]:
```
poll_data = 7
```

[86]:
```
type(poll_data)
```

[86]:
```
int
```

[88]:
```
set(range(9))
```

[88]:
```
{0, 1, 2, 3, 4, 5, 6, 7, 8}
```

[89]:
```
list(range(9))
```

[89]:
```
[0, 1, 2, 3, 4, 5, 6, 7, 8]
```

[90]:
```
tuple(range(9))
```

[90]:
```
(0, 1, 2, 3, 4, 5, 6, 7, 8)
```

[91]:
```
dict(range(9))
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[91], line 1
----> 1 dict(range(9))

TypeError: cannot convert dictionary update sequence element #0 to a sequence
```

[99]:
```
obj_data=()
```

[100]:
```
type(obj_data)
```

[100]:
```
tuple
```

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]: